Artificial and Computational Intelligence

AIMLCZG557

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

ACI Team

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

AIMLCZG557 – CS#6

Local Search Algorithms, Evolutionary Algorithms and Neural Architecture Search (NAS)

Agenda for CS #6

1) Recap of CS #5
2) Online Search Agents
3) Genetic Algorithms
4) Introduction to ACO
5) Neural Architecture Search
    1) Why?
    2) What?
    3) CoDeepNEAT
    4) Example
6) Q&A!

Online vs Offline Search

- **So far, most search algorithms assume an offline setting**
  - In offline search, the agent has a complete model of the problem before acting.
- **Offline search means:**
  - The full state space is known or can be generated in advance.
  - The agent can simulate possible actions before taking them.
  - A complete plan is produced first, then executed.
  - Examples: Breadth-First Search, Uniform Cost Search, A*, Hill Climbing, Local Beam Search.
- **Example**
  - A route-planning system knows the full map, road connections, and costs before suggesting a path.

Online Search Agents

- **Online search agents do not know the full environment in advance.**
  - They must act, observe the result, update their knowledge, and continue searching.
- **Online search means:**
  - The agent discovers the environment while moving through it.
  - It may not know all possible states or action outcomes initially.
  - It must balance exploration and progress toward the goal.
  - Planning and execution happen together.
  - Useful when the environment is unknown, dynamic, or too large to fully model.
- **Example**
  - A robot exploring an unfamiliar building cannot plan the full route in advance. It moves, senses nearby paths, updates its map, and decides the next action.

Genetic Algorithms

- A genetic algorithm (or GA) is a search technique used in computing to find true or approximate solutions to optimization and search problems.
- (GA)s are categorized as global search heuristics.
- (GA)s are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover (also called recombination).
- The evolution usually starts from a population of randomly generated individuals and happens in generations.
- In each generation, the fitness of every individual in the population is evaluated, multiple individuals are selected from the current population (based on their fitness), and modified to form a new population.

Genetic Algorithms

- The new population is used in the next iteration of the algorithm.
- The algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.
- No convergence rule or guarantee

Genetic Algorithms Vocabulary

- **Population** - Group of all individuals
- **Individual** - Any possible solution
- **Fitness** – Target function that we are optimizing (each individual has a fitness)
- **Trait** - Possible aspect (features) of an individual
- **Genome** - Collection of all chromosomes (traits) for an individual.

Genetic Algorithms

Population:

- Bio: Collection of Individuals.
- Algorithm: Collection of states.

Fitness:

- Bio: More healthy, less prone to diseases.
- Algorithm: Closest to the final solution.

Selection:

- Bio: Selecting species that are the most biological fit.
- Algorithm: Selecting states that are closest to the solution (fittest).

Crossover:

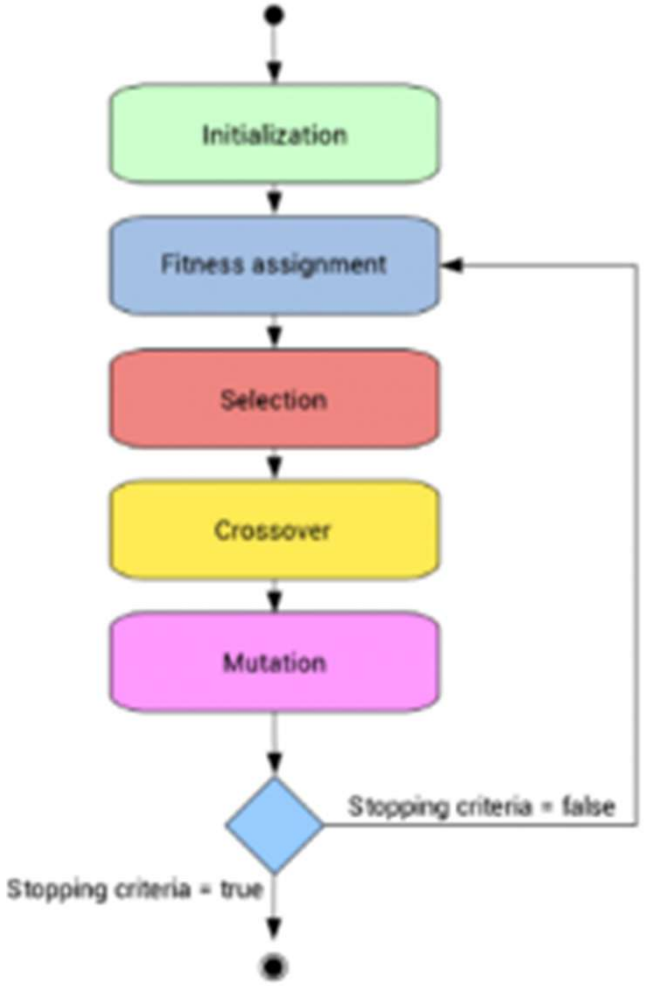
- Bio: Mating or reproducing
- Algorithm: Interchanging values between selected states

Mutation:

- Bio: Change or variation in DNA
- Algorithm: Alternation of state (randomly)

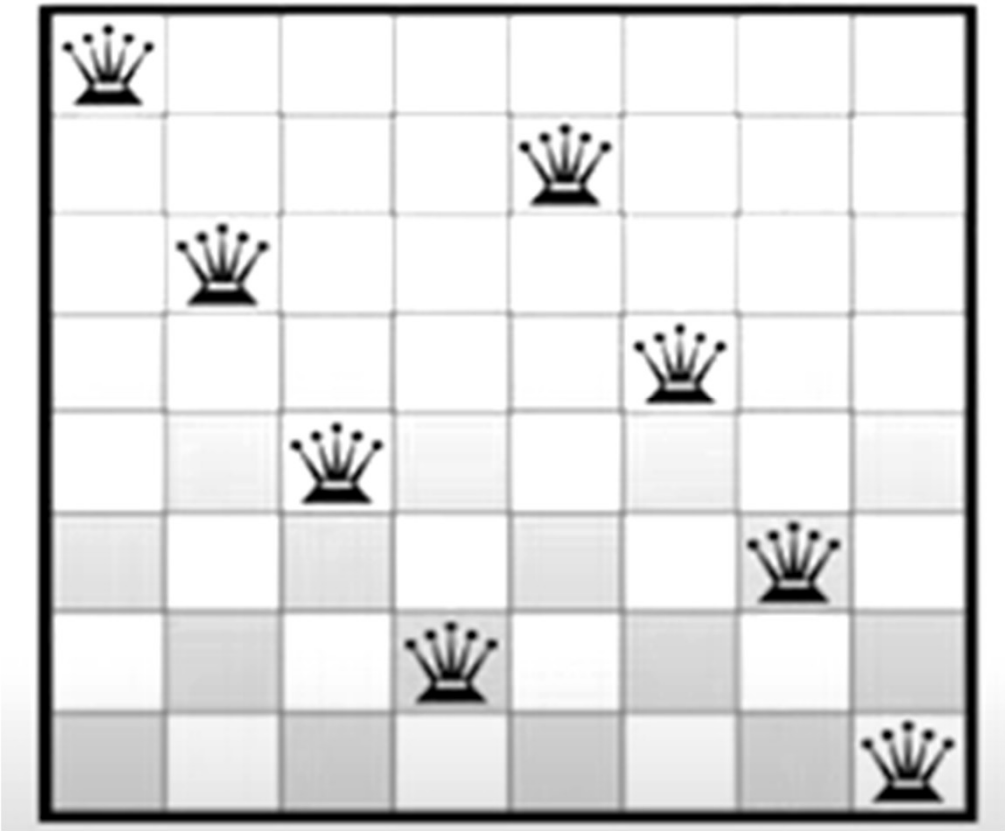
Basic Genetic Algorithm

- Start with a large "population" of randomly generated "attempted solutions" to a problem
- Repeatedly do the following:
  - Evaluate each of the attempted solutions
  - (probabilistically) keep a subset of the best solutions
  - Use these solutions to generate a new population
  - Quit when you have a satisfactory solution (or you run out of time)



8-Queens Problem

- Arrange 8 queens in such a way that no queen attacks each other.

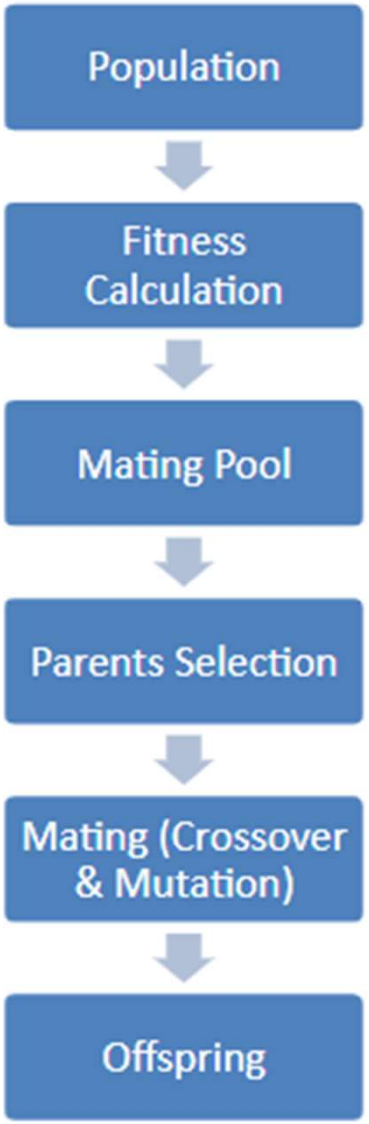


Solving 8-Queens Problem using Genetic Algorithm

- Step 01: Representing individuals
- Step 02: Generating an initial population.
- Step 03: Applying a fitness function.
- Step 04: Selecting parents for mating in accordance to their fitness.
- Step 05: Crossover of parents to produce new generation
- Step 06: Mutation of new generation to bring diversity
- Step 07: Repeat until solution is reached.

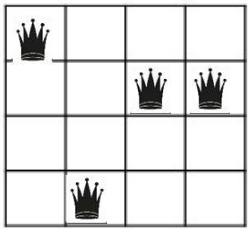
Genetic Algorithm

- Step 01: Representing individuals
- Step 02: Generating an initial population.
- Step 03: Applying a fitness function.
- Step 04: Selecting parents for mating in accordance to their fitness.
- Step 05: Crossover of parents to produce new generation
- Step 06: Mutation of new generation to bring diversity
- Step 07: Repeat until solution is reached.



Step 01: Representing Individuals

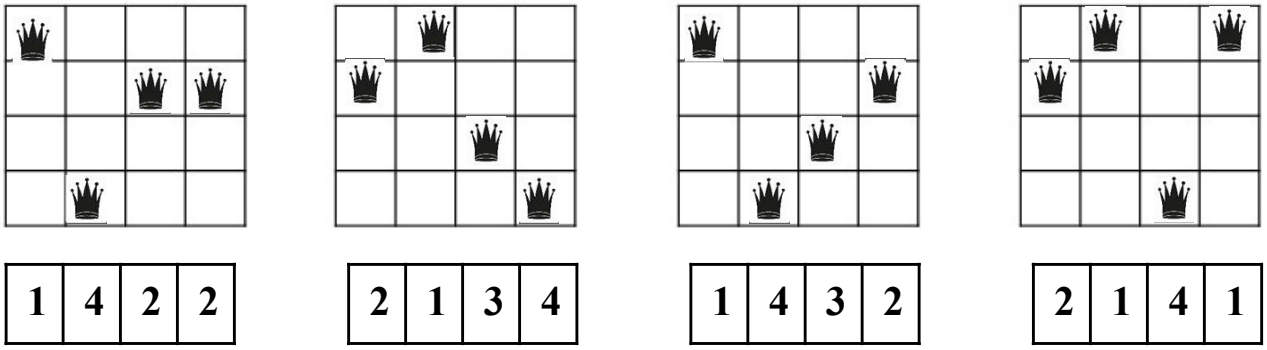
1. Select ‘k’ random states – Initialization : k=4
2. Evaluate the fitness value all states
3. If anyone of the state’s has achieved the threshold fitness value or threshold new states or no change is seen than previous iteration then the algorithm stops
4. Else, use roulette wheel mechanism to select pair/s
5. Pairs selected produces new state (successor) by crossover
6. Successor is allowed to mutate
7. Repeat from Step 2



1	4	2	2

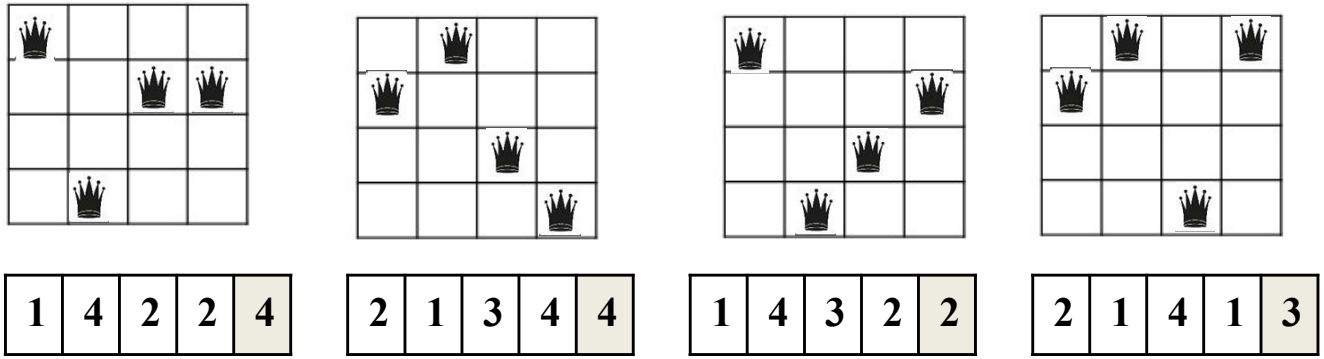
Step 02: Generate initial population

1. Select ‘k’ random states – **Initialization : k=4**
2. Evaluate the fitness value all states
3. If anyone of the state’s has achieved the threshold fitness value or threshold new states or no change is seen than previous iteration then the algorithm stops
4. Else, use roulette wheel mechanism to select pair/s
5. Pairs selected produces new state (successor) by crossover
6. Successor is allowed to mutate
7. Repeat from Step 2



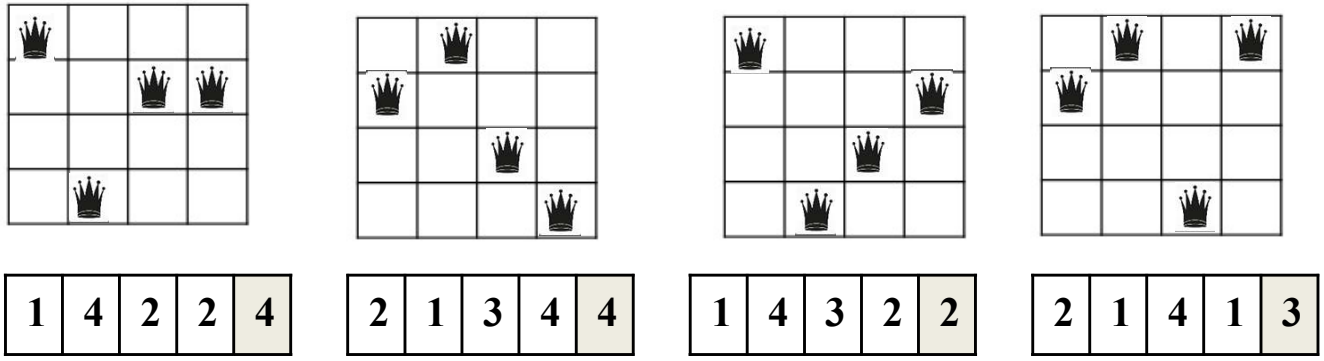
Step 03: Apply Fitness Function

1. Select ‘k’ random states – **Initialization : k=4**
2. Evaluate the fitness value all states : Maximizing function : No of Non-attacking pairs Queens → Threshold = 6
3. If anyone of the state’s has achieved the threshold fitness value or threshold new states or no change is seen than previous iteration then the algorithm stops
4. Else, use roulette wheel mechanism to select pair/s
5. Pairs selected produces new state (successor) by crossover
6. Successor is allowed to mutate
7. Repeat from Step 2



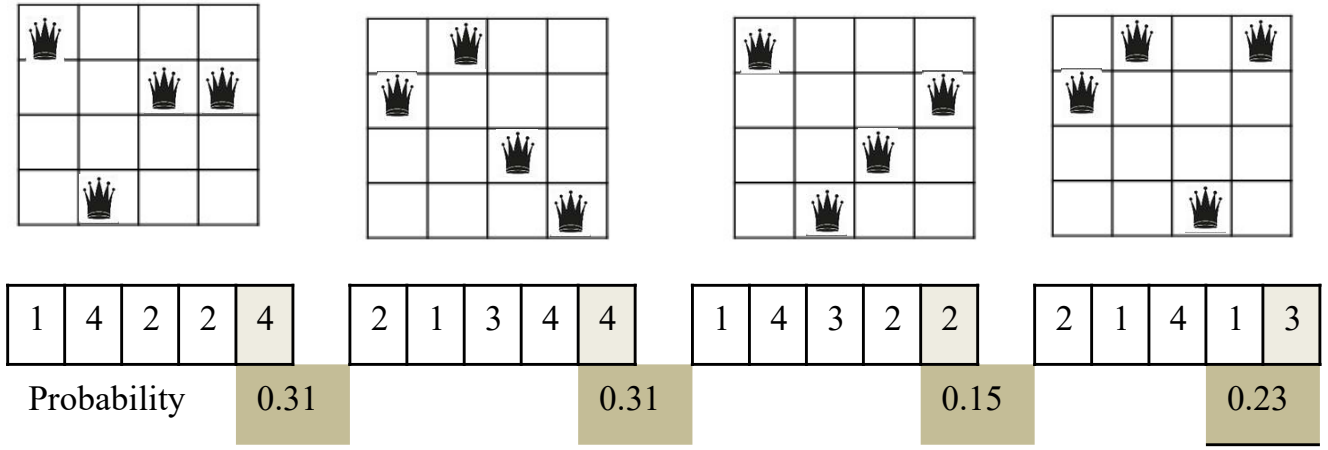
Step 03: Apply Fitness Function

1. Select ‘k’ random states – **Initialization : k=4**
2. Evaluate the fitness value all states : Maximizing function : No of Non-attacking pairs Queens → Threshold = 6
3. If anyone of the state’s has achieved the threshold fitness value or threshold new states or no change is seen than previous iteration then the algorithm stops
4. Else, use roulette wheel mechanism to select pair/s
5. Pairs selected produces new state (successor) by crossover
6. Successor is allowed to mutate
7. Repeat from Step 2



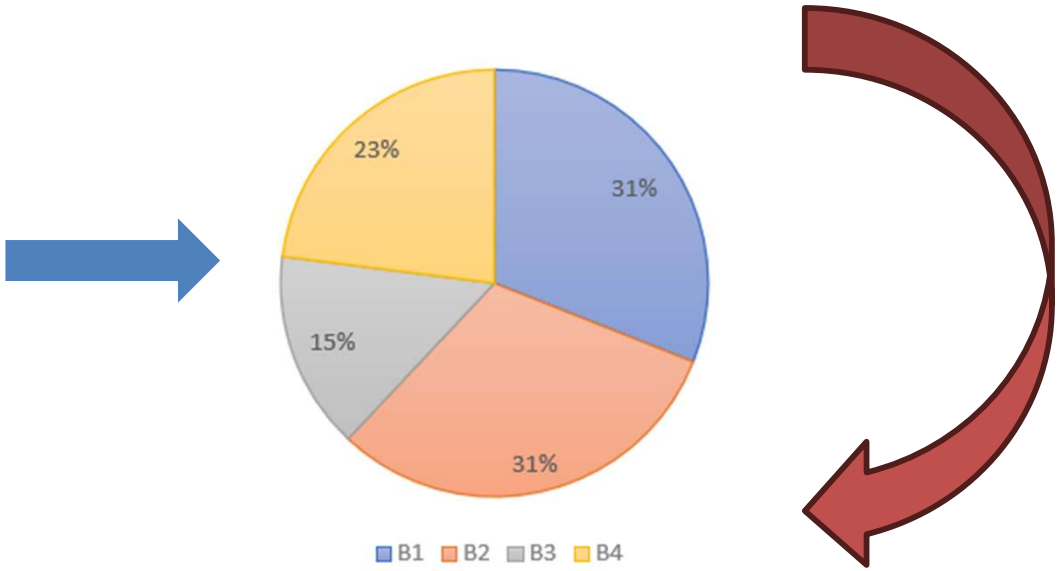
Step 04: Selection

1. Select ‘k’ random states – **Initialization : k=4**
2. Evaluate the fitness value all states : Maximizing function : No of Non-attacking pairs Queens → Threshold = 6
3. If anyone of the state’s has achieved the threshold fitness value or threshold new states or no change is seen than previous iteration then the algorithm stops
4. Else, use roulette wheel mechanism to select pair/s
5. Pairs selected produces new state (successor) by crossover
6. Successor is allowed to mutate
7. Repeat from Step 2



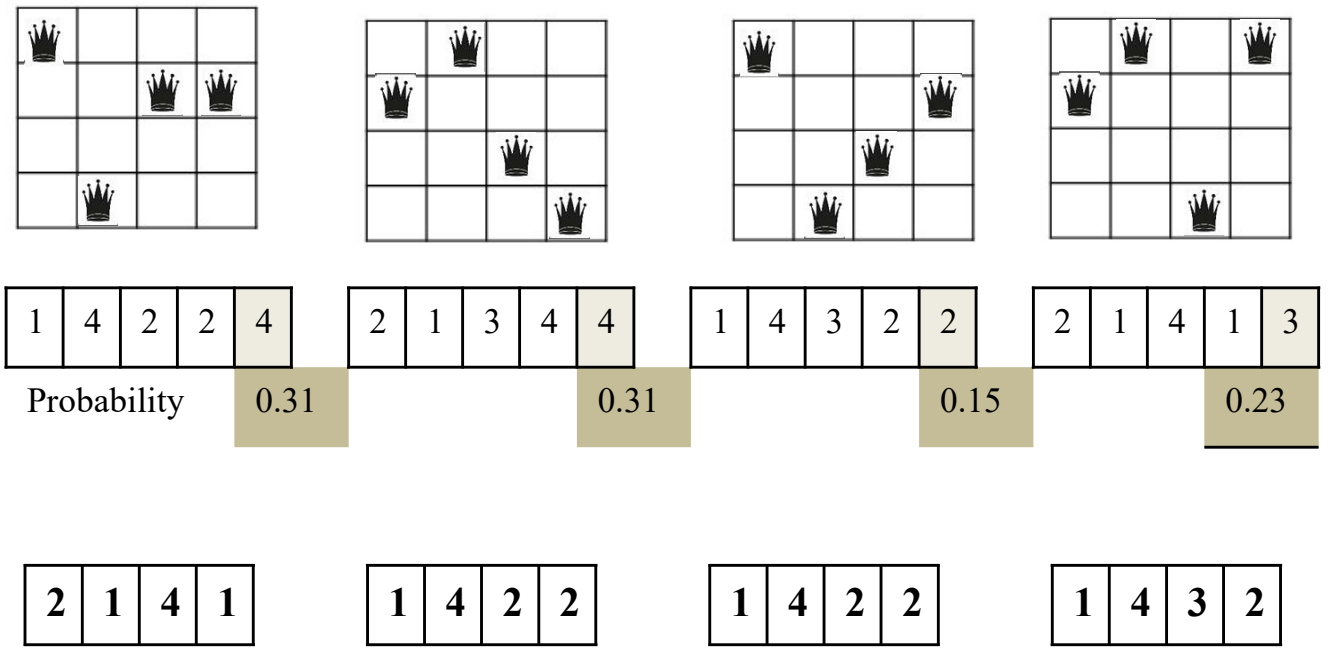
Step 04: Selection

- There are various methods of selection.
- Roulette wheel, tournament, rank, etc.
- Population is divided on a wheel according to their respective percentages of fitness and two fixed points are placed.
- Wheel is spun and those individual are selected at which the fixed points are pointing when the wheel stops.



Step 04: Selection

Sample winners of game:1,2,3,4 : B4, B1, B1, B3
So, <B4,B1> and <B1, B3> are the selected parents.



BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Crossover Strategies

There are several possible crossover Strategies:

- Randomly select a single point for a crossover
- Multi point crossover
- Uniform crossover

Single point crossover

Two point crossover (Multi point crossover)

Crossover Strategies

There are several possible crossover Strategies:

- Randomly select a single point for a crossover
- Multi point crossover
- Uniform crossover

Uniform crossover

- A random subset is chosen
- The subset is taken from parent 1 and the other bits from parent 2.

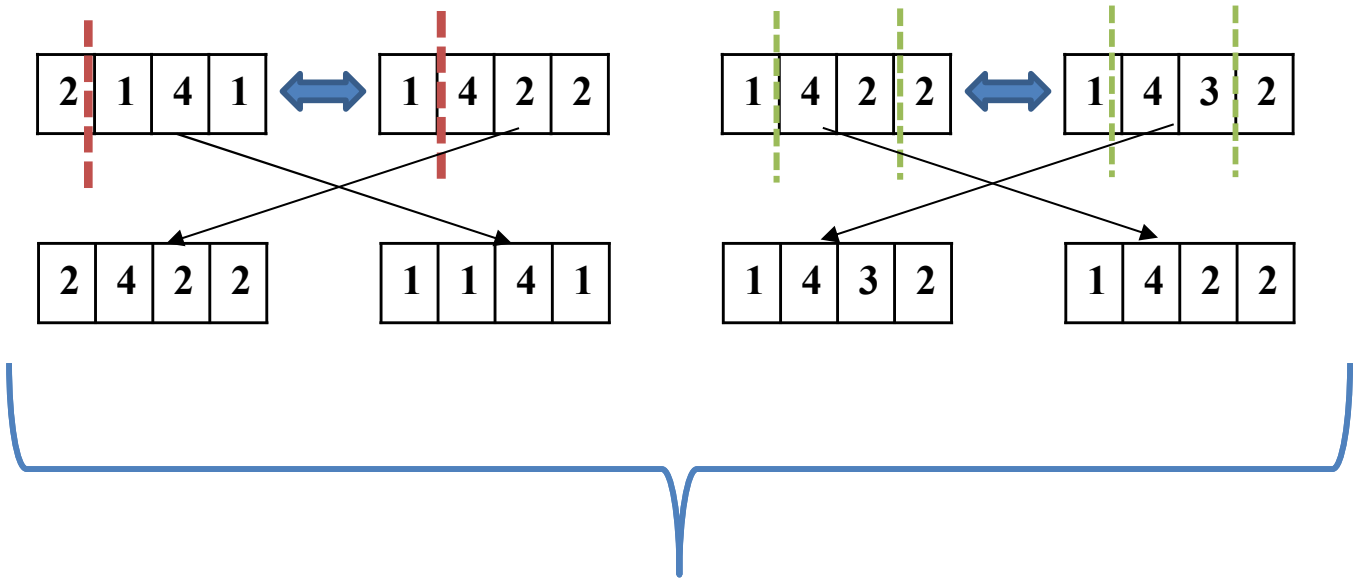
Subset: **BAABBAABBB** **(Randomly generated)**

Parents: **1010001110** **0011010010**

Offspring: **0011001010** **1010010110**

Step 05: Crossover

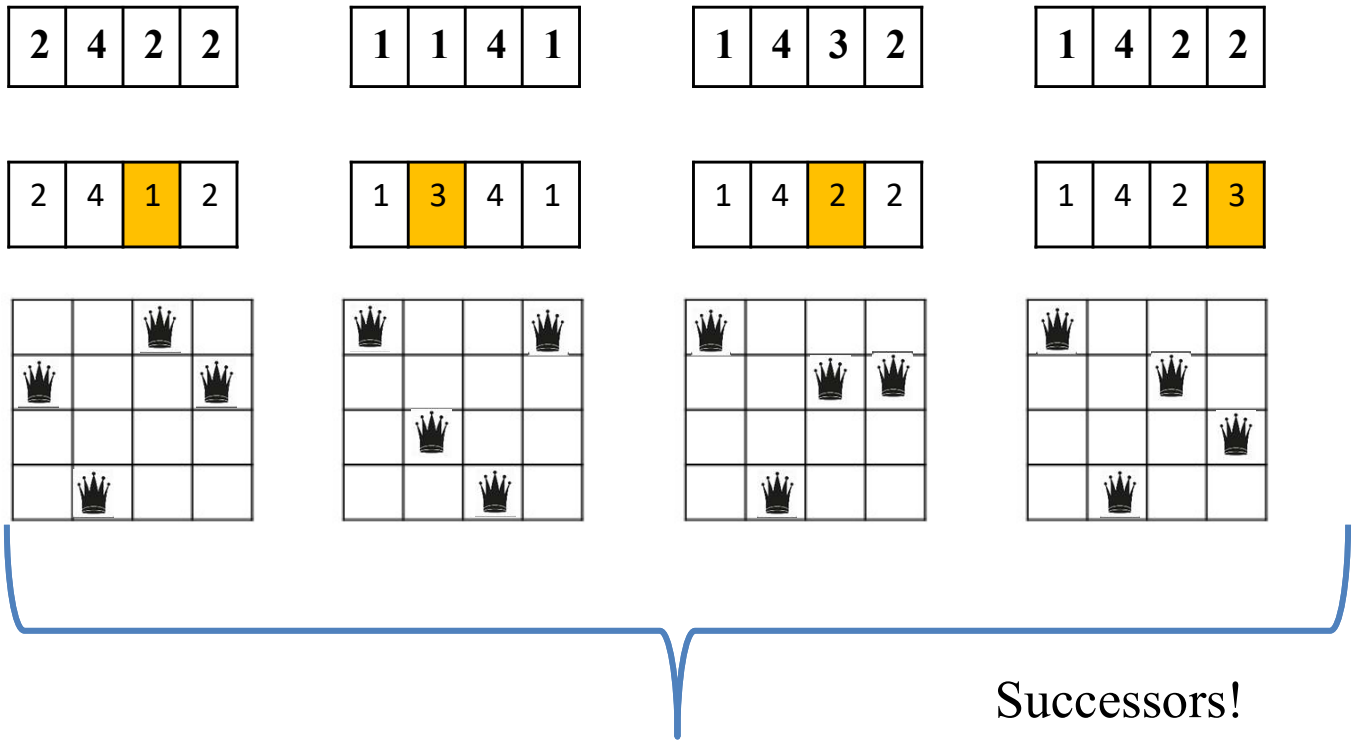
1. Select ‘k’ random states – **Initialization : k=4**
2. Evaluate the fitness value all states : Maximizing function : No of Non-attacking pairs Queens → Threshold = 6
3. If anyone of the state’s has achieved the threshold fitness value or threshold new states or no change is seen than previous iteration then the algorithm stops
4. Else, use roulette wheel mechanism to select pair/s
5. Pairs selected produces new state (successor) by crossover
6. Successor is allowed to mutate
7. Repeat from Step 2



Successors!

Step 06: Mutation

1. Select ‘k’ random states – **Initialization : k=4**
2. Evaluate the fitness value all states : Maximizing function : No of Non-attacking pairs Queens → Threshold = 6
3. If anyone of the state’s has achieved the threshold fitness value or threshold new states or no change is seen than previous iteration then the algorithm stops
4. Else, use roulette wheel mechanism to select pair/s
5. Pairs selected produces new state (successor) by crossover
6. Successor is allowed to mutate (Change Gene)
7. Repeat from Step 2

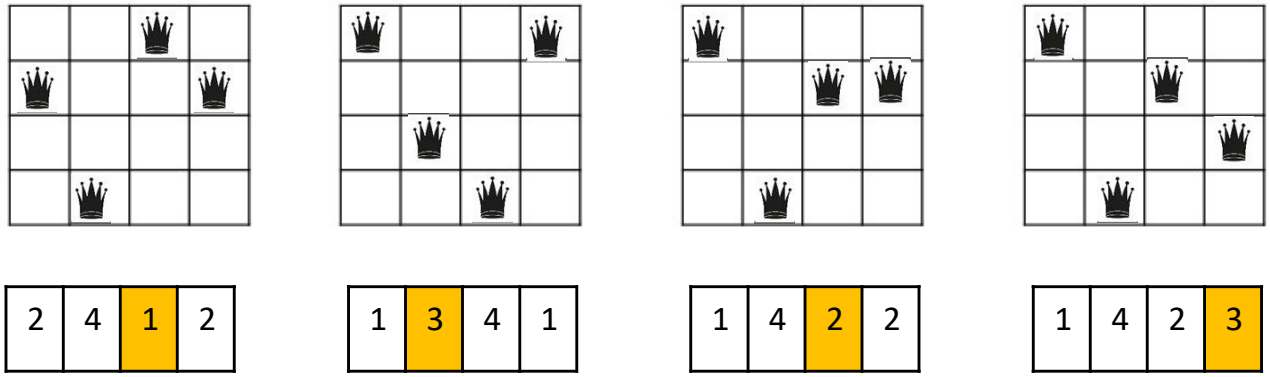


Iteration 1 Completed

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Step 07: Repeat!

1. Select ‘k’ random states – **Initialization : k=4**
2. Evaluate the fitness value all states : Maximizing function : No of Non-attacking pairs Queens → Threshold = 6
3. If anyone of the state’s has achieved the threshold fitness value or threshold new states or no change is seen than previous iteration then the algorithm stops
4. Else, use roulette wheel mechanism to select pair/s
5. Pairs selected produces new state (successor) by crossover
6. Successor is allowed to mutate (Change Gene)
7. Repeat from Step 2



- Calculate their fitness, if anyone has fitness as 6, STOP. Else
- Calculate probabilities, apply roulette wheel, perform cross over and mutation!

Genetic Algorithm

function GENETIC-ALGORITHM(*population*, FITNESS-FN) **returns** an individual
  inputs: *population*, a set of individuals
    FITNESS-FN, a function that measures the fitness of an individual

  repeat
    new_population ← empty set
    for $i = 1$ **to** SIZE(*population*) **do**
      x ← RANDOM-SELECTION(*population*, FITNESS-FN)
      y ← RANDOM-SELECTION(*population*, FITNESS-FN)
      child ← REPRODUCE(x, y)
      if (small random probability) **then** *child* ← MUTATE(*child*)
      add *child* to *new_population*
    population ← *new_population*
  until some individual is fit enough, or enough time has elapsed
  return the best individual in *population*, according to FITNESS-FN

---

function REPRODUCE(x, y) **returns** an individual
  inputs: x, y, parent individuals

  n ← LENGTH(x); c ← random number from 1 to n
  return APPEND(SUBSTRING($x, 1, c$), SUBSTRING($y, c + 1, n$))

Genetic Algorithm

Techniques:

1. Design of the fitness function
2. Diversity in the population to be accounted
3. Randomization

Application:

- Creative tasks
- Exploratory in nature
- Planning problem
- Static Applications

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Exercise 1

- Suppose local beam search has k = 3 and the following current states:

Current State	Heuristic Value
S1	12
S2	15
S3	10

- Their successors are:

Successor	Heuristic Value
A	16
B	11
C	14
D	18
E	13
F	17

If the goal is to maximize, which three states will be selected for the next beam?

Exercise 2

1. Given two parent chromosomes:

- Parent 1: [1, 2, 3, 4, 5, 6, 7, 8]
  Parent 2: [8, 7, 6, 5, 4, 3, 2, 1]
- Using single-point crossover after the 4th position, generate two children.

2. Suppose the fitness values of four individuals are:

Individual	Fitness
A	10
B	30
C	40
D	20

Which individual has the highest chance of being selected? Why?

Ant Colony Optimization

General pseudo-code

Procedure *ACO*

Schedule Activities

Initialization

Construction

Update Pheromone

Daemon Actions {optional}

// local search, elitism

End schedule activities

End *ACO*

Parameters used in ACO

Parameter	Description
N	Total No of ants ; N>1
τ_o	Initial pheromone amount
τ_{ij}	Amount of pheromone deposited while traversing from i to j
η_{ij}	1/(Cost of link (i,j))
α	Importance coefficient of pheromone intensity
β	Importance coefficient of route cost
ρ	evaporation co-efficient; $0<\rho<1$
$visit_k$	Visited nodes table of k^{th} ant
Q	Importance - Constant value pertaining to pheromone trail
$\boldsymbol{f_k}$	Route cost obtained by ant k

ACO Steps

Initialization

- Place predefined number of ants on starting point
- Set values for parameters α, β, ρ.
- Set τ_0 to 0.

– Construction

- Compute the next node transition probability

$$NTP_{ij} = \frac{(\tau_{ij})^{\alpha} (\eta_{ij})^{\beta}}{\sum_{h \notin visit_k} (\tau_{ih})^{\alpha} (\eta_{ih})^{\beta}}$$

ACO Steps

Pheromone updation:

- Pheromone reinforcement & pheromone evaporation
- Direct impact on the exploitation (enhancing found food path) & exploration (discovering new path) of ant algorithms

$$\tau_{ij}^{new} = (1-\rho)\tau_{ij}^{old} + \Delta\tau_{ij}^{k}$$

- Amount of pheromone deposited on (i,j) by kth at that timestamp is given by

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{\boldsymbol{f_k}} & if\ k^{th} ant\ passes\ i\ and\ j \\ 0 & otherwise \end{cases}$$

- **Stopping criteria:** reaching predetermined number of iterations

Problem: Reaching pre-determined number of iterations before reaching destination leading to ant drop

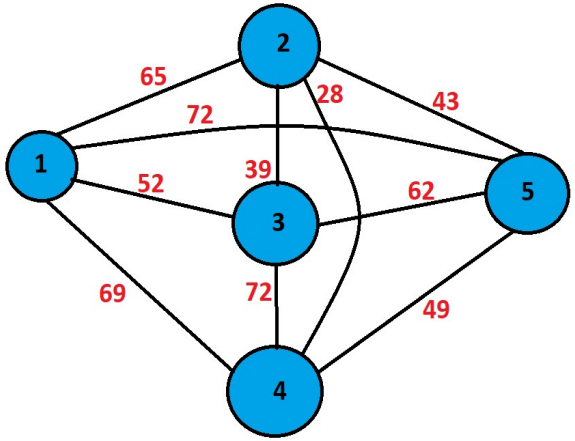
Travelling Salesman Problem

Problem: Given n cities, the goal is to find shortest path going through all cities and visiting each exactly once

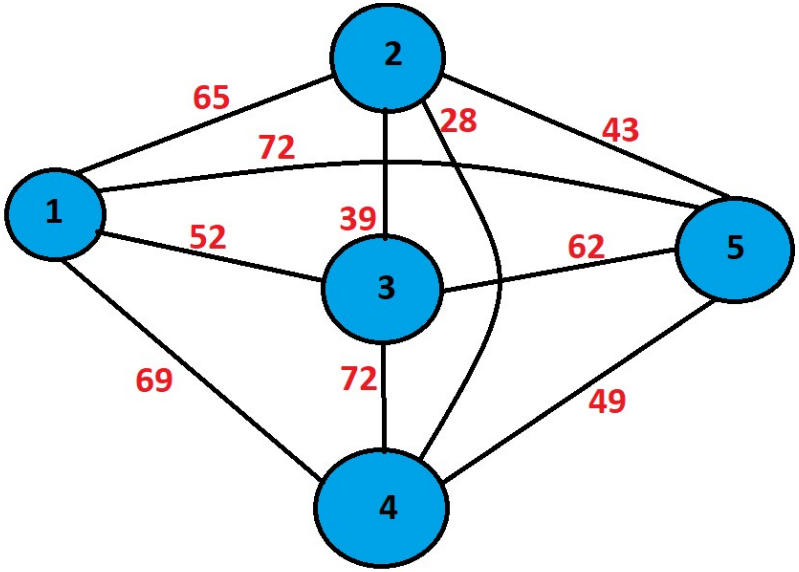
- Consider a complete graph

- d_{ij} is the route cost over (i,j) $\{f_k\}$

- Each ant builds its own tour from starting city
- Each ant chooses a town to go to with a probability
- Keep tabs on visit list of each ant
- When tour completed, lay pheromone on each edge visited
- Next city *j* after city *i* chosen according to probability rule



Travelling Salesman Problem using ACO



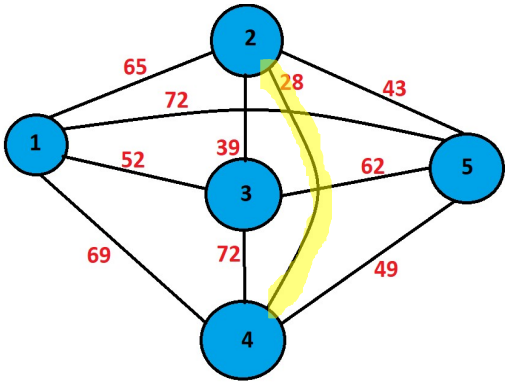
No. of ants = No. of cities = 5
Start at 4

$\alpha = 0.5$
$\beta = 0.75$
$\rho = 0.1$
$Q = 100$

$\tau_{12} = \tau_{21} = 0.54$
$\tau_{13} = \tau_{31} = 0.53$
$\tau_{14} = \tau_{41} = 0.35$
$\tau_{15} = \tau_{51} = 0.24$
$\tau_{23} = \tau_{32} = 0.53$
$\tau_{24} = \tau_{42} = 0.39$
$\tau_{25} = \tau_{52} = 0.18$
$\tau_{34} = \tau_{43} = 0.32$
$\tau_{35} = \tau_{53} = 0.9$
$\tau_{45} = \tau_{54} = 0.68$

	1	2	3	4	5
1	0	65	52	69	72
2	65	0	39	28	43
3	52	39	0	72	62
4	69	28	72	0	49
5	72	43	62	49	0

Step 1:
Next Transition Probability



$$NTP_{ij} = \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{n \in \text{visit}_k}(\tau_{in})^{\alpha}(\eta_{in})^{\beta}}$$

$$NTP_{41} = \frac{(\tau_{41})^{\alpha}(\eta_{41})^{\beta}}{(\tau_{41})^{\alpha}(\eta_{41})^{\beta} + (\tau_{42})^{\alpha}(\eta_{42})^{\beta} + (\tau_{43})^{\alpha}(\eta_{43})^{\beta} + (\tau_{45})^{\alpha}(\eta_{45})^{\beta}}$$

$$= \frac{(0.35)^{0.5} \cdot (1/69)^{0.75}}{(0.35)^{0.5} \cdot (1/69)^{0.75} + (0.39)^{0.5} \cdot (1/28)^{0.75} + (0.32)^{0.5} \cdot (1/72)^{0.75} + (0.68)^{0.5} \cdot (1/49)^{0.75}}$$

$$= \frac{(0.35)^{0.5} \cdot (0.0145)^{0.75}}{(0.35)^{0.5} \cdot (0.0145)^{0.75} + (0.39)^{0.5} \cdot (0.0357)^{0.75} + (0.32)^{0.5} \cdot (0.0139)^{0.75} + (0.68)^{0.5} \cdot (0.0204)^{0.75}}$$

$$= \frac{0.0247}{0.1434}$$

$P_{41} = 0.1723$

$$P_{42} = \frac{(0.39)^{0.5} \cdot (1/28)^{0.75}}{0.1434} = \frac{0.0513}{0.1434}$$

$P_{42} = 0.3577$

$$P_{43} = \frac{(0.32)^{0.5}(\frac{1}{72})^{0.75}}{0.1434} = \frac{0.0229}{0.1434}$$

$P_{43} = 0.1596$

$$P_{45} = \frac{(0.68)^{0.5}(\frac{1}{49})^{0.75}}{0.1434} = \frac{0.0445}{0.1434}$$

$P_{45} = 0.3104$

P_{42} is max. Therefore, move from 4 to 2.

Pheromone Updation

$$\tau_{ij}^{\text{new}} = (1-\rho)\tau_{ij}^{\text{old}} + \Delta\tau_{ij}^{k}$$

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k} & \text{if ant } k \text{ passes edge } ij \\ 0 & \text{otherwise} \end{cases}$$

Since ant moved from 4 to 2, calculate $\Delta\tau_{12}$:

$$\tau_{12} = (1-0.1)(0.54) + 0 = 0.486$$

$$\tau_{13} = (1-0.1)(0.53) + 0 = 0.477$$

$$\tau_{14} = (1-0.1)(0.35) + 0 = 0.315$$

$$\tau_{15} = (1-0.1)(0.24) + 0 = 0.216$$

$$\tau_{23} = (1-0.1)(0.53) + 0 = 0.477$$

$$\tau_{24} = \tau_{42} = (1-0.1)(0.39) + \frac{100}{28} = 3.924$$

$$\tau_{25} = (1-0.1)(0.16) + 0 = 0.162$$

$$\tau_{34} = (1-0.1)(0.32) + 0 = 0.288$$

$$\tau_{35} = (1-0.1)(0.9) + 0 = 0.81$$

$$\tau_{45} = (1-0.1)(0.68) + 0 = 0.612$$

Step 2:
Next Transition Probability

innovate achieve lead

$Visit_k = 4 - 2$

Find P_{21}, P_{23} and P_{25}

$$P_{21} = \frac{(0.486)^{0.5}\left(\frac{1}{65}\right)^{0.75}}{(0.486)^{0.5}\left(\frac{1}{65}\right)^{0.75} + (0.477)^{0.5}\left(\frac{1}{39}\right)^{0.75} + (0.162)^{0.5}\left(\frac{1}{43}\right)^{0.75}}$$

$$= \frac{(0.6971)(0.0437)}{(0.6971)(0.0437) + (0.6907)(0.064) + (0.4025)(0.0596)}$$

$$P_{21} = \frac{0.0305}{0.0987}$$

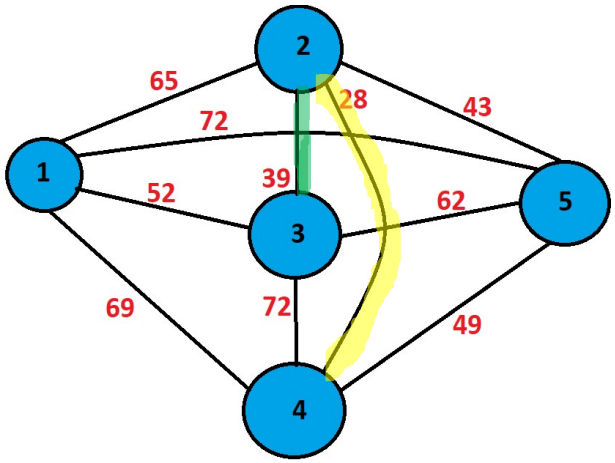
$P_{21} = 0.3086$

$$P_{23} = \frac{0.0443}{0.0987}$$

$P_{23} = 0.4485$

$$P_{25} = \frac{0.0240}{0.0987}$$

$P_{25} = 0.2429$



P_{23} is maximum. Now ant moves from 2 to 3.

Pheromone Updation

$$\tau_{ij}^{\text{new}} = (1-\rho)\tau_{ij}^{\text{old}} + \Delta\tau_{ij}^{k}$$

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k} & \text{if ant } k \text{ passes edge } ij \\ 0 & \text{otherwise} \end{cases}$$

Since ant moved from 2 to 3, calculate $\Delta\tau_{23}$:

$$\tau_{12} = (1-0.1) \times 0.486 + 0 = 0.4374$$

$$\tau_{13} = (1-0.1) \times 0.477 + 0 = 0.4293$$

$$\tau_{14} = (1-0.1) \times 0.315 + 0 = 0.2835$$

$$\tau_{15} = (1-0.1) \times 0.216 + 0 = 0.1944$$

$$\tau_{23} = (1-0.1) \times 0.477 + \frac{100}{39} = 2.9934$$

$$\tau_{24} = (1-0.1) \times 3.9224 + 0 = 3.5302$$

$$\tau_{25} = (1-0.1) \times 0.162 + 0 = 0.1458$$

$$\tau_{34} = (1-0.1) \times 0.288 + 0 = 0.2592$$

$$\tau_{35} = (1-0.1) \times 0.810 + 0 = 0.729$$

$$\tau_{45} = (1-0.1) \times 0.612 + 0 = 0.5508$$

Step 3:
Next Transition Probability

innovate achieve lead

$Visit_k = 4 - 2 - 3$

Find P_{31} and P_{35}

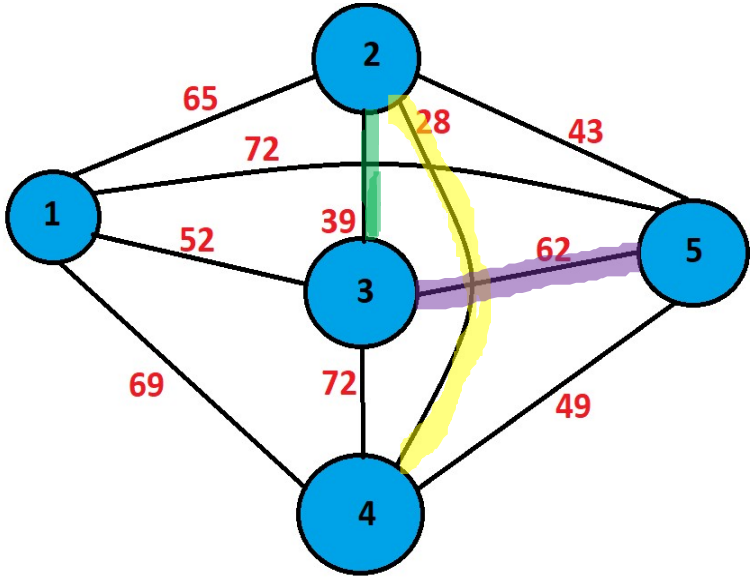
$$P_{31} = \frac{(0.4293)^{0.5}\left(\frac{1}{52}\right)^{0.75}}{(0.4293)^{0.5}\left(\frac{1}{52}\right)^{0.75} + (0.729)^{0.5}\left(\frac{1}{62}\right)^{0.75}}$$

$$= \frac{0.0338}{0.0725}$$

$P_{31} = 0.4668$

$$P_{35} = \frac{0.0386}{0.0725}$$

$P_{35} = 0.5332$



P_{35} is maximum. Now ant moves from 3 to 5.

Pheromone Updation

$$\tau_{ij}^{\text{new}} = (1-\rho)\tau_{ij}^{\text{old}} + \Delta\tau_{ij}^{k}$$

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k} & \text{if ant } k \text{ passes edge } ij \\ 0 & \text{otherwise} \end{cases}$$

Since ant moved from 3 to 5, calculate $\Delta\tau_{35}$:

$$\tau_{12} = (1-0.1) \times 0.4374 + 0 = 0.3937$$

$$\tau_{13} = (1-0.1) \times 0.4293 + 0 = 0.3864$$

$$\tau_{14} = (1-0.1) \times 0.2835 + 0 = 0.2552$$

$$\tau_{15} = (1-0.1) \times 0.1944 + 0 = 0.1750$$

$$\tau_{23} = (1-0.1) \times 2.9334 + 0 = 2.6941$$

$$\tau_{24} = (1-0.1) \times 3.5302 + 0 = 3.1772$$

$$\tau_{25} = (1-0.1) \times 0.1458 + 0 = 0.1312$$

$$\tau_{34} = (1-0.1) \times 0.2592 + 0 = 0.2333$$

$$\tau_{35} = (1-0.1) \times 0.729 + \frac{100}{62} = 2.269$$

$$\tau_{45} = (1-0.1) \times 0.5508 + 0 = 0.4957$$

Step 4:
Next Transition Probability

innovate achieve lead

$Visit_k = 4 - 2 - 3 - 5 - 1$

Now ant moves from 5 to 1.

Pheromone Updation:

$$\tau_{12} = 0.9 \times 0.3937 + 0 = 0.3543$$

$$\tau_{13} = (1 - 0.1) \times 0.3864 + 0 = 0.3477$$

$$\tau_{14} = (1 - 0.1) \times 0.2552 + 0 = 0.2296$$

$$\tau_{15} = (1 - 0.1) \times 0.175 + \frac{100}{72} = 1.5464$$

$$\tau_{23} = (1 - 0.1) \times 2.6941 + 0 = 2.4247$$

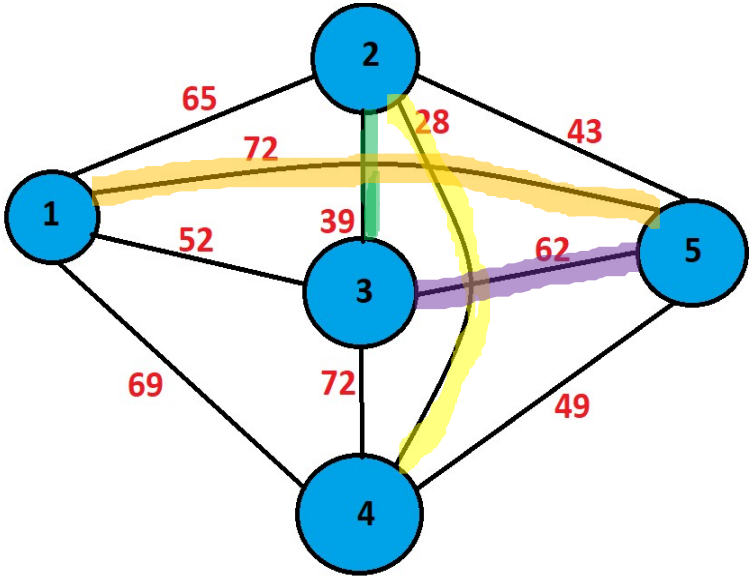
$$\tau_{24} = (1 - 0.1) \times 3.1772 + 0 = 2.8595$$

$$\tau_{25} = (1 - 0.1) \times 0.1312 + 0 = 0.1181$$

$$\tau_{34} = (1 - 0.1) \times 0.2333 + 0 = 0.21$$

$$\tau_{35} = (1 - 0.1) \times 2.269 + 0 = 2.0421$$

$$\tau_{45} = (1 - 0.1) \times 0.4957 + 0 = 0.4461$$



Step 5:
Next Transition Probability

innovate achieve lead

$Visit_k = 4 - 2 - 3 - 5 - 1 - 4$

Now ant moves from 1 to 4 (origin).

Pheromone Updation:

$$\tau_{12} = (1 - 0.1) \times 0.3543 + 0 = 0.3189$$

$$\tau_{13} = (1 - 0.1) \times 0.3477 + 0 = 0.313$$

$$\tau_{14} = (1 - 0.1) \times 0.2296 + \frac{100}{69} = 1.6559$$

$$\tau_{15} = (1 - 0.1) \times 1.5464 + 0 = 1.3917$$

$$\tau_{23} = (1 - 0.1) \times 2.4247 + 0 = 2.1822$$

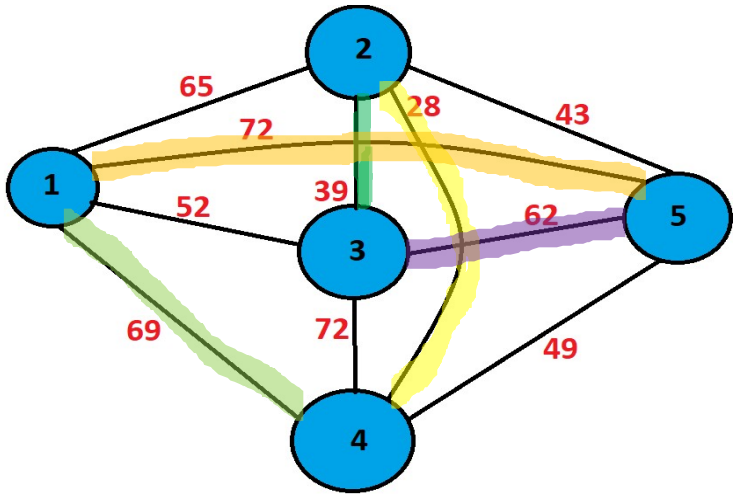
$$\tau_{24} = (1 - 0.1) \times 2.8595 + 0 = 2.5735$$

$$\tau_{25} = (1 - 0.1) \times 0.1181 + 0 = 0.1063$$

$$\tau_{34} = (1 - 0.1) \times 0.21 + 0 = 0.189$$

$$\tau_{35} = (1 - 0.1) \times 2.0421 + 0 = 1.8379$$

$$\tau_{45} = (1 - 0.1) \times 0.4461 + 0 = 0.4015$$



Module 3 : Problem Solving Agent using Search

A. Informed Search Algorithms ✅

B. Heuristic Functions ✅

C. Local Search Algorithms & Optimization Problems ✅

D. Neural Architecture Search

The Era of Self-Designing AI

- CoDeepNEAT (Co-evolving Deep NeuroEvolution of Augmenting Topologies) is an *automated* method for designing deep neural network architectures.
- Traditional AI relies on human experts to manually design architectures; CoDeepNEAT replaces this with evolutionary search.
- Inspired by natural evolution, architectures are gradually improved through selection, mutation, and recombination.
- Goal: Automatically evolve a neural network that can accurately classify images such as “Cat” vs “Dog” without manual design decisions.

The Human Bottleneck

- Designing deep neural networks manually is challenging due to a vast search space of possible architectures and hyperparameters.
- Human experts rely on trial-and-error and prior knowledge, which limits exploration of novel architectures.
- As network complexity increases, manual optimization becomes inefficient and impractical.
- CoDeepNEAT addresses this by evolving a population of architectures and selecting the most effective ones based on performance.

DNA vs. Weights (The Hybrid Optimization Approach)

CoDeepNEAT follows a bilevel optimization strategy combining evolution and learning.

1. Architecture (DNA / Topology): Designed using evolutionary algorithms (high-level search).
2. Weights (Learned Parameters): Optimized using gradient descent during training (low-level learning).
    - Evolution explores structural design, while learning optimizes performance within that structure.
    - This synergy enables automatic discovery of architectures that are both effective and efficient.

What is Neuroevolution?

Neuroevolution = Using genetic algorithms to evolve neural networks

How It Works (Core Loop)

1. Create many neural networks (population)
2. Train each network
3. Measure performance (fitness)
4. Select best networks
5. Generate new networks using:
    - Mutation
    - Crossover
6. Repeat

- **Population** → Set of networks
- **Fitness** → How good the network performs
- **Generation** → One cycle of evolution

Understanding NEAT

- NEAT stands for NeuroEvolution of Augmenting Topologies.
- It is a method that evolves neural networks automatically instead of designing them manually.
- A neural network is represented as a graph. Nodes represent neurons and edges represent connections between them.
- It starts with very simple networks having minimal connections. Over generations, it gradually adds new nodes and connections to increase complexity.
- Each new connection is assigned a unique innovation number. These numbers help match corresponding genes correctly during crossover.
- Similar networks are grouped into species based on their structure. Each species evolves separately, maintaining diversity in the population.

Limitation of NEAT

- NEAT performs well for small and simple neural networks but is not effective for deep learning models with many layers.
- Modern AI systems use deep architectures such as CNNs and LSTMs, which require layer-wise design and complex structures.
- NEAT evolves individual neurons and connections, but not full layered architectures.
- Therefore, there is a need for methods that can evolve deep neural networks.
- This limitation led to advanced approaches like DeepNEAT and CoDeepNEAT.

We will learn game play in the next class …

Exercise 1

A warehouse robot begins at C1, tasked with collecting Items (It) and reaching the Dock (DV) at C4 on a 4x4 grid. The robot must navigate hazards like blocked cells (#), Spills (SP) with high movement costs, while leveraging Energy packs (E) to reduce costs. Using the A* algorithm, the robot calculates the optimal path by considering both actual path cost and a heuristic. The agent has four distinct actions: Move Up, Move Down, Move Left, and Move Right. However, it cannot move into blocked cells. The warehouse is represented as a 4×4 grid below:

	1	2	3	4
A	It	E		SP
B	SP	It	E	SP
C	S	E	It	DV
D	It	SP	SP	E

Entering SP: +12; collecting It: −8 (reward); energy pack E: +1.
Heuristic h(n) = Manhattan(robot, DV) + (# remaining Items) + (# adjacent SP to robot in resultant state).

a) Expand and depict the search tree for exactly 2 levels. (Level 0 = Start).
b) Calculate Path costs and heuristics (g/h/f) for all generated nodes.
c) Apply A* search and show all the steps.

Exercise 2

1. Given two parent chromosomes:

- Parent 1: [1, 2, 3, 4, 5, 6, 7, 8]
  Parent 2: [8, 7, 6, 5, 4, 3, 2, 1]
- Using single-point crossover after the 4th position, generate two children.

2. Suppose the fitness values of four individuals are:

Individual	Fitness
A	10
B	30
C	40
D	20

Which individual has the highest chance of being selected? Why?

ACI: Disclaimer and Acknowledgements

- Few content for these slides may have been obtained from prescribed books and various other source(s) on the Internet.
- We hereby acknowledge all the contributors for their material and inputs and gratefully acknowledge people others who made their course materials freely available online. We have provided source information wherever necessary.
- We have added and modified the content to suit the requirements of the class dynamics & live session's lecture delivery flow for presentation.
- This *is not a full-fledged* reading materials. Students are requested to refer to the textbook w.r.t detailed content as per the course handout as shared over e-learning portal - Taxila.
- **Slide Source / Preparation / Review:**
- From BITS Pilani WILP**:** Prof. Rajavadhana, Prof. Indumathi, Prof. Sangeetha and Prof Parthasarathy PD, Prof Ramya Devi
- From BITS On-campus & External : Mr.Santosh GSK

Thank You for your time & attention !

Slides are Licensed Under : CC BY-NC-SA 4.0